ÇANKAYA UNIVERSITY
FACULTY OF ENGINEERING

CENG 408

Innovative System Design and Development Ⅱ
Project Report

Spaceborn

Aleyna Saykılı 201911054

Aslı Ceren Hızır 202128035

Dilara Bayındır 202011079

Ata Doruk Çakmak 202111030

Oğuz Akay 202128011

Advisors: Murat SARAN- Talha KARADENİZ

Table of Contents

1. Introduction

This document provides a comprehensive overview of the software design of the Spaceborn project. It is a technical guide aimed at clarifying the project, system architecture, technical components and data. Spaceborn is a survival and colony management game set in space, providing management of game resources, building systems and protecting colonies from external threats. The sections in this document cover advanced features and usage scenarios, explaining the general storage and storage of the system step by step. The literature review provides background information supporting the goals of Spaceborn by introducing the basic collection and comprehensive features. The Software Requirements Specification (SRS) section clarifies the goals, parameters and general features of the game. The Software Design Document (SDD) details the architecture, features and design elements that are deployed from the user perspective. This document provides a comprehensive guide for presenting and explaining the Spaceborn project step by step.

2. Project Plan

SPACEBORN WORK TABLE

DEADLINES	11.10. 2024	18.10. 2024	25.10. 2024	08.11. 2024	06.12. 2024	13.12. 2024	27.12. 2024	10.01. 2025	???
Team Setup									
Project Selection Form									
Github Repository									
Project Work Plan									
Literature Review									
SRS Document									
Project Webpage									
SDD Document									
Project Report									
Presentation									

3. Literature Review

3.1 öz

Spaceborn projesi, uzay temalı bir koloni inşa etme ve hayatta kalma oyunu olarak tasarlanmıştır. Bu oyun strateji ve puzzle temalarını bir araya getirerek oyuncuları içine çekmeyi hedefler. Oyuncular prosedürel olarak oluşturulmuş bir uzay gemisinde kolonilerin hayati sistemlerini kontrol ederken bir yandan da düşman mutantlara karşı savunma yapar. Oyuncu kolonilerin hayati sistemini onların oksijen seviyesini, gıda ihtiyaçlarını ve enerji yönetimlerini dengeleyerek sağlar. Ayrıca prosedürel harita sayesinde de oyuncu her oynayışta farklı deneyimler elde edebilir. Bu oyunu geliştirebilmek için Unreal Engine oyun motoru kullanılacaktır. Kodlama için C++ dilinin performans avantajlarından ve Unreal Engine'nin görsel programlama aracı olan blueprint ile hızlı oyun mekaniği oluşturma avantajından yararlanılacaktır. Ek olarak tasarım ve modellemeler için Blender ve Photoshop uygulamaları kullanılacaktır. Projeyi geliştirirken teknik riskler, kaynak yetersizliği ve en önemlisi market riski göz önünde bulundurulacaktır. Bunun için piyasadaki benzer oyunlar araştırılıp onlardan farklı olarak sunabileceğimiz oyun mekaniği, oyun tasarımı ve oyunun hikayesi üzerine durulacaktır.

3.2 Abstract

The Spaceborn project is designed as a space-themed colony building and survival game. This game aims to captivate players by combining strategy and puzzle elements. Players manage the vital systems of colonies aboard a procedurally generated spaceship while defending against hostile mutants. The player ensures the colony's vital systems by balancing oxygen levels, food needs, and energy management. Additionally, thanks to the procedural map, players can experience different challenges with each playthrough. This game will be developed using the Unreal Engine game engine. The performance advantages of C++ will be utilized for coding, while the visual programming tool, Blueprint, will be leveraged to quickly create game mechanics. Additionally, Blender and Photoshop will be used for design and modeling. While developing the project, technical risks, resource shortages, and, most importantly, market risks will be taken into account. To address this, similar games in the market will be researched, and the focus will be on offering unique game mechanics, game design, and storyline elements that differentiate the game from others.

Introduction

The gaming industry has become a rapidly growing sector in recent years. With the advancement of technology, games have become more visually impressive and gameplay has become smoother. Divided into branches such as mobile, PC, and console games, it appeals to a wide range of age groups. Every day, new technologies are being developed in this field, allowing it to continue evolving. Strategy games hold an important place among video games. They provide players with opportunities for planning, making tactical decisions, and critical thinking. Players are required to think not only in the short term but also in the long term. Especially with the advancement of technology, artificial intelligence and procedural generation have made the game more dynamic, offering players a different experience with each playthrough.

The Spaceborn project is a game that invites players to establish a colony in space. In this game, the player must meet the essential needs for the colony's survival. Additionally, they must combat mutants that have infested the spaceship and face various challenges. These challenges include system malfunctions, resource shortages, and health crises within the colony. The game features procedurally generated maps and events, encouraging players to explore new sections of the spaceship and nearby space objects to gather rare resources. This ensures that the game offers a strategy-themed experience and provides a unique playthrough each time. The goal of the game is to create an enjoyable experience that offers strategy and immersive gameplay for fans of sci-fi and survival genres. The core elements of strategy games are as follows: 1.Resource Management: It requires the player to plan how to use limited resources most efficiently. In our game, resource management is done by adjusting the colony's oxygen levels, food needs, and energy levels. By managing these resources, the player ensures the colony's survival. 2.Planning and Timing: Players must make long-term plans and implement them at the right time. In our game, the player must plan how to meet the colony's food needs and ensure that the necessary actions for survival are carried out in time. While doing this, they must also remember the hostile mutants. 3.Defense and Attack Mechanics: Players must develop defense strategies and attack enemies to slow them down or neutralize them. In our game, there will be three different types of enemies. The player must develop defense and attack strategies based on these enemy types.

4. Used Technologies

Unreal Engine, developed by Epic Games, is a powerful game engine. It will be used as the main game engine for the development of the Spaceborn project. Unreal Engine has attracted the attention of developers due to its high-quality graphics. With C++ integration and Unreal Engine's visual programming tool, Blueprint, game mechanics, animations, AI systems, and UI components are being developed. C++ provides performance advantages for developers, while Blueprint offers fast prototyping and debugging processes. For the design aspects of the game, we will utilize Blender and Photoshop. Blender is a tool used for 3D modeling, animation, and visual effects development. It will be used in the Spaceborn project for environmental design, the design of the game character and enemy characters, as well as for creating their in-game animations. Photoshop is used for 2D assets. It will be used for menu designs, icon designs, and environmental designs within the game. With these tools, we will create a visually impressive game with dynamic animations. Git is a version control system commonly used in game development projects. By using this system, multiple developers can work simultaneously, have the ability to back up and revert code. In our project, Git version control will be used, allowing the team to work together while tracking the development and changes of the project. Additionally, it will offer the ability to easily revert back if any issues arise. Procedural generation algorithms will be used to randomly generate maps and events. AI systems will also be implemented to code NPC behaviors and mechanics, providing a dynamic experience for the player.

4.1 Risks

1.Technical Risks: The biggest technical challenges in the project are the complexity of procedural generation and AI behaviors. Procedural generation of the map may cause errors in the systems to be developed. In addition, additional errors may occur depending on the unexpected behavior of the player. Apart from this, performance problems may occur. When using high and quality visuals, we need to consider system requirements in terms of performance. In order to reduce these risks, we will start the development process with the procedural map first and try to solve possible errors in advance. In addition, we aim to detect and solve possible problems that may occur early by testing the game at every stage.

2.Resource Risks: The most important resource risk in our game is that team members have insufficient knowledge on some subjects. For example, in our

project, only one person is familiar with the animation and modeling part, but for a game, there are too many components to be modeled and animated, so we will try to solve this problem by getting support from ready-made assets. In addition, artificial intelligence and procedural generation algorithms are also difficult topics for developers. Team members will receive cross-training for these and these gaps will be closed. Another resource risk is that not enough games are tested because the more a game is tested, the more we observe possible errors in advance. We plan to solve this problem by holding a test event in our school and having school students play it.

3.Marketing Risks:

The most important market risk in a game project is the intensity of competition. There are many survival and strategy games on the market, and some games are very well-established and good games. Your game needs to differentiate itself from these games in some way. We aim to offer players a new style of play by creating different game mechanics than the games on the market. We also plan to differentiate the game by offering players a different story from other strategy games with the story of the game. Determining the wrong target audience is also very important in the marketing part of the game. We aim to determine this after doing market research and present it to the right audience. Another marketing risk is receiving negative criticism and reviews. We believe that we can solve this by testing before the game is released and by getting feedback from people.

4.2 Existing/Similar Solutions

1.Subnautica

Solution Description: "Subnautica" is a sci-fi survival and exploration game where players find themselves stranded on an oceanic alien planet. The game focuses on survival elements such as managing food, water, and oxygen while exploring the deep sea, gathering resources, crafting equipment, and uncovering the planet's mysteries.

Shortcomings: "Subnautica" is based on an open-world ocean planet environment, lacking the confined space and colony protection dynamics of a spaceship. In contrast, "Spaceborn" offers a colony-building experience in a limited space, emphasizing resource management within the constraints of a spaceship, as well as defense against mutant threats.

2.Dead Space

Solution Description: "Dead Space" is a sci-fi horror game where players navigate a spaceship overrun by hostile alien-like creatures. It emphasizes a story-driven survival horror experience, with players managing limited resources and battling mutants to survive in a dark, atmospheric environment.

Shortcomings: While "Dead Space" delivers a compelling horror experience, it lacks colony-building, resource management, and open-ended exploration elements. "Spaceborn" differentiates itself by integrating survival, resource management, and a colony-defense strategy into a narrative-driven gameplay, combining elements of survival horror with sci-fi colony management.

3.Frostpunk

Solution Description: "Frostpunk" is a survival and colony simulation game where players manage resources and citizens' needs to survive in a post-apocalyptic frozen wasteland. The game requires strategic decision-making to keep a growing city alive in the face of environmental threats and resource scarcity.

Shortcomings: "Frostpunk" is focused on city-building in a harsh environment on Earth and lacks the unique atmosphere of a space-based setting. "Spaceborn" takes the colony management concept into space, presenting unique challenges within a spaceship environment and combining it with exploration of space objects and defensive mechanics against mutant threats.

4.Alien: Isolation

Solution Description: "Alien: Isolation" is a sci-fi survival horror game where players must avoid a terrifying alien while managing resources and navigating a hostile spaceship. It combines stealth and horror elements in a highly atmospheric environment.

Shortcomings: "Alien: Isolation" is heavily focused on horror and stealth but lacks colony management, resource-building, and defense setup mechanics. In "Spaceborn," players not only defend against threats but also manage resources, set up defenses, and construct systems essential for the survival of a colony, blending survival horror with strategic elements.

5.RimWorld

Solution Description: "RimWorld" is a sci-fi colony simulator where players manage colonists, resources, and defenses while responding to randomly generated events on an alien planet. It focuses on colony management, resource allocation, and responding to environmental threats.

Shortcomings: Although "RimWorld" provides a colony management experience, it is planet-based rather than confined to a spaceship environment. "Spaceborn" offers the added challenge of limited space within a ship, as well as the threat of mutated entities, creating a more intense and strategic survival experience with space exploration opportunities.

5. Conclusion

The "Spaceborn" project offers a unique blend of survival, strategy, and combat mechanics within a confined space-based setting, providing an alternative to existing games by combining resource management, defense, and exploration with a story-driven approach. During the development process, game dynamics and visual elements will be designed effectively with a powerful game engine such as Unreal Engine, performance-oriented languages such as C++, and design tools such as Blender and Photoshop. The "Spaceborn" project, taking into account potential risks and market research, will provide an exciting experience for fans of sci-fi and strategy games.

Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed overview of the "Spaceborn" project, including its objectives, requirements, technological specifications, risks, and an analysis of existing solutions. It is intended to guide the project development team and stakeholders in understanding and aligning on the goals and direction of the project.

1.2 Scope

1.2.1 Software Product Identification

The software product to be developed is called "Spaceborn". It is a single-player survival and puzzle-based game designed to be played on PCs. The game integrates various systems, including resource management mini-games, AI-driven alien threats, and procedural map generation, to create a unique survival experience in a confined spaceship setting.

1.2.2 Software Product Capabilities

What the software will do:

• Resource Management: Players will solve mini-games (e.g., puzzles for oxygen restoration, power rerouting) to maintain vital spaceship systems and ensure the survival of human embryos onboard.

• Alien Encounters: The game features multiple types of alien threats, including an intelligent machine-learning-driven enemy that adapts and attempts to track the player.

• Procedural Map Generation: Each gameplay session will have a uniquely generated spaceship layout to enhance replayability.

• Solo Experience: The game is designed for single-player use, optimized for smooth performance on mid-range systems.

• Immersive Storyline: Players uncover the backstory of the mutants, the spaceship's mission, and their ultimate objective while trying to survive until help arrives.

What the software will not do:

• The game will not include multiplayer or online functionality.

• It will not feature external modifications or integrations beyond the core gameplay mechanics.

1.2.3 Application and Benefits

Application of the Software:

"Spaceborn" immerses players in a high-stakes survival experience where they must rely on critical thinking, problem-solving, and strategic decision-making to manage resources and avoid alien threats.

Relevant Benefits, Objectives, and Goals:

• Replayability: Procedural map generation and dynamic alien AI behaviors create a unique experience with every playthrough.

• Engaging Gameplay: Mini-games add depth to resource management, making routine survival tasks interactive and challenging.

• Cognitive Challenge: The machine-learning-powered enemy adds unpredictability, testing the player's adaptability and decision-making under pressure.

• Optimization: Designed to deliver a smooth, visually stunning experience even on mid-range systems, increasing accessibility.

• Story-Driven Engagement: A compelling narrative keeps players motivated and emotionally invested in the survival of the characters and their mission.

1.2.4 Consistency with Higher-Level Specifications

• The description aligns with the overarching system requirements, including the integration of mini-games for resource management, the use of Unreal Engine (C++) for development, and Python for machine learning implementation. These ensure the game meets its objectives of delivering an optimized, engaging, and replayable survival experience.

• Similar products such as "Frostpunk" and "Subnautica" serve as inspirations, demonstrating how resource management and survival can be intricately tied to player engagement. Additionally, games like "Alien: Isolation" showcase the tension and unpredictability of being hunted by adaptive AI, a feature reflected in the machine learning-driven alien encounters in "Spaceborn." Lastly, the procedural generation mechanics align with approaches used in games like "No Man's Sky", ensuring each session offers unique and fresh gameplay experiences.

1.3 Glossary

A table of definitions, terms, and abbreviations used in this document

Term	Description
UI	User Interface
AI	Artificial Intelligence
FPS	First Person Shooter
HUD	Heads-Up Display

1.4 Overview

Spaceborn combines strategy, defense, and story-driven gameplay to provide players with a space survival experience. Unlike existing survival games, the game takes place within a confined ship environment and encourages exploration of new procedurally generated sections with space objects.

1.5 Version History

Provide a table of table of changes to this document

Version No	Description of change	Date
1.0	Initial Release	15.10.2024

2. Overall Description

2.1 Product Perspective

Spaceborn is an independent, self-contained software product designed as a single-player survival game. The product is not a component of a larger system and operates as a standalone application. However, it relies on integration between Unreal Engine, C++, and Python for core functionalities such as game mechanics, procedural map generation, and machine-learning-driven alien behaviors.

The game operates under various constraints and interfaces to provide an optimized and seamless gaming experience. A block diagram is included to illustrate the major components of the system, their interconnections, and interfaces.

2.1.1 System Interfaces

Spaceborn will be developed using either Unity or Unreal Engine and will be compatible with the following systems:

• Operating Systems: Windows and macOS, with future plans for Linux compatibility.

• Hardware Requirements: Mid-range gaming PCs with sufficient resources to support real-time rendering and procedural generation.

• Game Engine: Unity: Known for its flexibility and user-friendly environment, Unity supports efficient procedural generation and 2D/3D asset integration. Unreal Engine: Offers high-quality rendering capabilities and advanced physics simulation for a visually immersive experience. Both engines provide the tools required to implement Spaceborn's core mechanics, including procedural map generation, AI systems, and user interfaces. The final decision will depend on the project team's expertise and the specific features prioritized during development.

2.1.2 User Interfaces

The user interface of Spaceborn will be developed using either Unity or Unreal Engine, depending on the final selection for the game engine. The design focuses on user-friendliness and simplicity to ensure an intuitive experience for players. The main menu includes options such as “New Game,” “Load Game,” “Settings,” and “Exit Game.” The Settings screen allows players to adjust parameters like “Music Volume,” “Resolution,” “Display Mode,” and “Control Sensitivity.” The in-game menu provides options for adjusting “Music Volume,” “Resolution,” and “Display Mode” while also featuring buttons for “Save,” “Cancel,” and “Quit Game.”

The gameplay screen is designed for clarity:

• The top of the screen displays critical resources such as oxygen, energy, and food levels.

• The bottom left features a timeline and a minimap for easy navigation and time management.

• Additional functionalities, such as resource allocation, construction, and defense options, are accessible through dedicated buttons located at the bottom of the screen.

• This interface ensures seamless navigation and interaction within the game, enhancing the player's immersion in the Spaceborn environment.

2.1.3 Hardware Interfaces

Spaceborn will utilize standard PC controls and hardware interfaces to provide an optimal gaming experience. The game will require the following:

• PC Devices: Spaceborn will run on devices with Windows or macOS. Future compatibility with Linux-based systems may also be considered.

Supported Devices: Mid-range PCs with the following configuration: Minimum: 8 GB RAM, quad-core processor, and GTX 1050 or equivalent GPU. Recommended: 16 GB RAM, RTX 3060 or higher GPU.

• Input Devices: The keyboard will be used for essential gameplay functions such as movement, shortcuts, and resource management. The mouse will enable navigation, object selection, and interaction with in-game menus and systems. Future updates may include support for gaming controllers for players who prefer this input method.

• Ports/Protocols: Game saves are stored locally; no external devices or connections are necessary.

2.1.4 Software Interfaces

• Operating System Compatibility:

Spaceborn will be compatible with major desktop operating systems, including Windows and macOS. The game engine, Unity or Unreal Engine, will provide a stable framework for UI creation, rendering, and core game mechanics.

• Procedural Generation: The chosen engine (Unity or Unreal Engine) will manage the procedural generation of maps and events to ensure unique gameplay experiences. Algorithms for randomization and resource placement will be seamlessly integrated.

• Physics Computations: Both Unity and Unreal Engine provide robust physics systems. These will be used to handle object collisions, environmental effects, and crew movement within the spaceship environment.

• Character and Object Animations: The animation tools in Unity or Unreal Engine will ensure smooth transitions and realistic behaviors for characters, objects, and enemies, creating an engaging and visually appealing game.

2.1.5 Communications Interfaces

Spaceborn operates offline and does not require network communications. However, communication protocols between the game engine and Python are facilitated locally through API calls or shared libraries.

2.1.6 Memory Constraints

Primary Memory: Requires a minimum of 8 GB of RAM for smooth gameplay.

Secondary Memory: The game requires approximately 10 GB of storage space for installation and save files.

2.1.7 Operations

Normal Operations:

• Players interactively solve puzzles, manage resources, and evade enemies.

Backup and Recovery:

• Autosave system triggers after completing major objectives, such as solving puzzles or escaping alien encounters.

• Manual save option available during safe zones.

2.1.8 Site Adaptation Requirements

Data Initialization: The procedural map generation requires predefined templates and initialization sequences specific to gameplay themes.

Adaptation Features: The game engine includes scalability settings, allowing it to adjust graphical fidelity based on the system's hardware capabilities.

2.2 Product Functions Procedural Map Generation:

• The game dynamically generates a unique spaceship layout for every playthrough, ensuring replayability and unpredictability.

Resource Management Mini-Games: Players interact with engaging mini-games to manage critical systems such as:

• Oxygen Supply: Maintaining breathable air for survival.

• Power Distribution: Allocating limited energy to different systems.

• Embryo Preservation: Keeping embryos stable and alive under resource constraints.

Machine Learning-Driven Alien AI:

• One of the alien enemies uses a machine learning algorithm to adapt its behavior based on player actions, creating an intelligent and evolving challenge.

Enemy Variants and Combat:

• Different types of alien creatures with distinct abilities attack the player, requiring strategic planning and fast reflexes to survive.

• Combat includes defensive tools, stealth mechanics, and environmental traps.

Survival Mechanics:

• Players must evade enemies while managing health and stress levels, creating a tense and immersive experience.

Progression and Story Unfolding:

• Completing resource tasks and evading threats gradually reveals the mysterious origins of the mutants and the true mission of the spaceship.

Save and Load System:

• Autosave functionality ensures progress is retained after critical events. Manual saves are available in safe zones.

Optimized Performance:

• The game ensures a smooth experience even on mid-range hardware by offering scalable graphics and efficient memory usage.

Logical Organization of Functions The game functions are interdependent:

• Resource Management directly impacts survival mechanics, as depleted resources increase vulnerability to alien threats.

• Procedural Map Generation creates a unique environment that dictates the placement of resources and enemy behavior.

• Machine Learning AI adjusts based on player strategies, making each playthrough a unique challenge.

2.3 User Characteristics

The intended users of Spaceborn include a diverse audience of gamers with varying levels of experience and technical expertise. The general characteristics of the target user base are as follows:

Educational Level:

• No specific educational qualifications are required, but users should possess basic literacy and comprehension skills to follow tutorials and in-game instructions.

Gaming Experience:

• Beginner to Intermediate Gamers: The game includes intuitive tutorials and guided objectives to accommodate users new to survival or resource management games.

• Experienced Gamers: Procedural maps and machine-learning-driven AI provide unique challenges for seasoned players seeking replayability and strategic depth.

Demographics:

• Age Group: 16 and above, as the game includes intense survival mechanics and suspenseful encounters that may not be suitable for younger players.

• Interest Areas: Fans of survival, science fiction, horror, and strategic resource management games are the primary audience.

Impact on Design and Requirements: The user characteristics influence several aspects of the game's design:

• Ease of Use: Simple and intuitive user interfaces, supported by an optional tutorial mode.

• Scalability: Difficulty levels allow beginners to enjoy the game while providing a challenge to experienced gamers.

• Accessibility: Optimized for mid-range systems to ensure a wide audience can access the game.

• Replayability: Procedural content and evolving AI provide engaging experiences for players seeking variety and depth.

2.4 Constraints The development of Spaceborn is subject to the following constraints:

Technological Constraints

• The game must be developed using either Unity or Unreal Engine, limiting the available development tools and requiring the team to work within the capabilities and limitations of the chosen engine. Procedural generation

algorithms and AI systems must be implemented efficiently to ensure real-time performance on mid-range gaming systems.

Resource Constraints

• The team has a limited number of members with specialized skills, such as 3D modeling, AI programming, and game design. Cross-training is required to ensure all critical tasks are addressed. Available hardware and software licenses may limit the tools and resources that can be utilized.

Temporal Constraints

• The project timeline is constrained to a fixed academic schedule, with specific milestones to be completed within weekly sprints and a final deadline for project submission.

Budget Constraints

• Financial resources are limited to available institutional or personal funding, restricting the ability to acquire premium software tools, high-end hardware, or external services.

Design Constraints

• The game must support procedurally generated maps and events while maintaining balanced gameplay and performance. It must also provide an engaging user interface and intuitive controls suitable for a broad audience.

Market Constraints

• The game faces competition from established titles with similar mechanics, such as RimWorld and Subnautica, necessitating a focus on unique features and a compelling gameplay experience. By understanding and addressing these constraints, the Spaceborn team aims to deliver a high-quality product within the available resources and timeline.

2.5 Assumptions and Dependencies

The following assumptions and dependencies are critical for the successful development of Spaceborn and may affect the requirements outlined in this SRS:

• Game Engine Availability and Selection: It is assumed that either Unreal Engine or Unity will be available for development throughout the project lifecycle. If, for any reason, the chosen engine is not available or fails to meet

the requirements for gameplay, performance, or machine learning integration, the project scope may need to be reassessed and the development timeline adjusted.

• Machine Learning Libraries and Python Integration: The game will rely on Python for machine learning integration (e.g., AI behavior and procedural generation). It is assumed that necessary libraries (such as TensorFlow, Keras, or PyTorch) will be compatible with the chosen game engine. Any limitations in integrating Python libraries with Unreal Engine or Unity could impact the design and AI features.

• System Requirements and Hardware Constraints: The game will be developed for mid-range gaming PCs, with the assumption that players will have a system meeting basic specifications (e.g., 8GB RAM, mid-range GPU). If the hardware capabilities change significantly during development, it may require adjustments in graphics, performance optimization, and the overall game experience.

• Third-Party Software and Tools: The project will use third-party tools for tasks like asset creation, procedural generation, and possibly sound effects. It is assumed that these tools will remain compatible with the chosen engine and that licensing costs will be affordable. Any changes to the availability, pricing, or compatibility of these tools could impact the development process.

• Procedural Generation Requirements: The assumption is that procedural generation will remain feasible and scalable within the constraints of the chosen game engine. If limitations arise in implementing a truly randomized experience, adjustments to the game's replayability and map design may be needed.

• Testing and Debugging Tools: It is assumed that suitable testing and debugging tools will be available and integrated with the development environment. If any tools become unavailable or incompatible, alternative solutions will need to be evaluated to maintain testing standards.

• Console Porting: The assumption is that once the PC version is stable, the game will be ported to consoles. This process will depend on the compatibility of the chosen engine with console platforms and the necessary adjustments for control schemes, performance, and hardware limitations. If console porting proves too complex or incompatible, the scope may be adjusted to focus solely on PC platforms.

• User Base and Feedback: It is assumed that the target user base for Spaceborn will remain consistent with the current market research, with a preference for survival and strategy games. Any significant shift in player expectations or trends may require changes in gameplay mechanics or feature prioritization.

2.6 Apportioning of Requirements

The following features and functionalities are identified as enhancements that may be included in future versions of Spaceborn. These are not critical to the core gameplay and can be delayed for later updates:

• Advanced Machine Learning for Enemy AI: Expanding the machine learning-driven AI to incorporate collaborative strategies among enemies or adapt to multiple playstyles more dynamically.

• Console Porting: Developing versions compatible with major gaming consoles, such as PlayStation, Xbox, and Nintendo Switch. This includes optimizing performance, adjusting controls, and ensuring compliance with platform-specific requirements.

• Game Engine Selection: The final decision between Unreal Engine and Unity will be delayed to a future phase. This allows for an evaluation of each engine's capabilities in terms of graphical fidelity, performance, and integration with machine learning and procedural generation systems.

3. Specific Requirements

This section outlines all the software requirements for the Spaceborn game, providing the necessary details to design, implement, and test the system. These requirements cover the inputs, outputs, and all functions the system must perform to meet the expectations of users and ensure successful gameplay.

3.1 External Interface Requirements

This section describes the inputs and outputs for Spaceborn. It includes data formats, interaction methods, and system responses to external stimuli.

3.1.1 Input and Output Overview:

Name of Item: Player Input (e.g., Movement, Interaction)

• Purpose: To allow players to control their character and interact with the game world.

• Source of Input: Player actions through keyboard/mouse/controller.

• Valid Range: Movement range within the game map (0–1000 units).

• Timing: Input is processed in real-time during gameplay.

• Relationships to Other Inputs/Outputs: Player movement affects the AI behavior, resource collection, and environmental changes.

Name of Item: AI Behavior Update (Mutants, Enemies)

• Purpose: To process AI decisions and update the behaviors of mutants and enemies based on player interactions.

• Source of Input: AI algorithms that analyze player position and actions.

• Output: Mutant movement, attacks, or strategy changes.

• Valid Range: Behavior can range from passive (avoidance) to aggressive (attack).

• Timing: AI behavior updates every 1-2 seconds, triggered by player interactions or environmental changes.

• Relationships to Other Inputs/Outputs: AI behavior is influenced by player actions and environmental conditions.

Name of Item: Save Game Data

• Purpose: To allow players to save and load their game progress.

• Source of Input: Player action (Save command).

• Output: A saved game file, including player inventory, AI states, and resource levels.

• Valid Range: Save game data must include up to 3 game slots for multiple save states.

• Timing: Save operation happens when the player manually saves or at checkpoints.

• Relationships to Other Inputs/Outputs: Saves include data from the current game session, which may include all player interactions and progress.

3.2 Functional Requirements

3.2.1 Save Game Functionality

• Introduction/Purpose of Feature: This feature allows players to save their current progress in Spaceborn, including game state, inventory, and resource levels.

• Stimulus/Response Sequence:

Stimulus: The player presses the "Save" button or triggers the save command from the menu.

Response: The system stores the game state (including player progress, resources, and AI states) and provides a confirmation message, such as "Save Successful."

• Associated Functional Requirements: The system shall save the player's progress, including inventory, resources, completed mini-games, and AI behaviors. The system shall provide multiple save slots for players to choose from and manage their progress.

3.2.2 AI Behavior Updates

• Introduction/Purpose of Feature: This feature ensures the AI behavior of mutants and enemies is updated based on player interactions and events in the game.

• Stimulus/Response Sequence:

Stimulus: Player actions such as movement, attacking, or interacting with certain game elements (e.g., triggering alarms).

Response: The AI system updates the mutants' behaviors, such as attacking, avoiding, or preparing for a counteraction based on the player's actions.

• Associated Functional Requirements:

The system shall update the AI behavior of mutants in real-time as the player interacts with the environment.

The system shall track and adjust the mutant's AI state based on factors such as health, aggression level, and location.

3.2.3 Resource Management

• Introduction/Purpose of Feature:

This feature manages the collection and consumption of resources (e.g., food, oxygen, energy) within the colony and the spaceship.

• Stimulus/Response Sequence:

Stimulus: The player interacts with the environment, collects, or uses resources. Response: The system updates the resource count and adjusts the colony's status accordingly (e.g., when resources are low, the game may prompt the player for action).

• Associated Functional Requirements:

The system shall track and update resource levels, including food, oxygen, and energy. The system shall alert the player when resources are running low or when certain thresholds are reached. requirements define the specific actions the system must perform in response to inputs and how it should process and generate outputs.

3.2.4 Main Menu Use Case

Brief Description:

The use case diagram outlines a main menu that provides users with several options to engage with the system. The main menu includes the following features: starting a new game, loading a saved game, accessing settings, and exiting the application.

Initial Step by Step Description:

1. New Game:When the player clicks on the "New Game" option, a fresh gaming session begins.
2. Load Game: Upon selecting the "Load Game" option, a list of previously saved game sessions is displayed.
3. Settings: Choosing the "Settings" option brings up a configuration panel for adjusting various game parameters.
4. Exit: Selecting the "Exit" option ends the current session or closes the application.

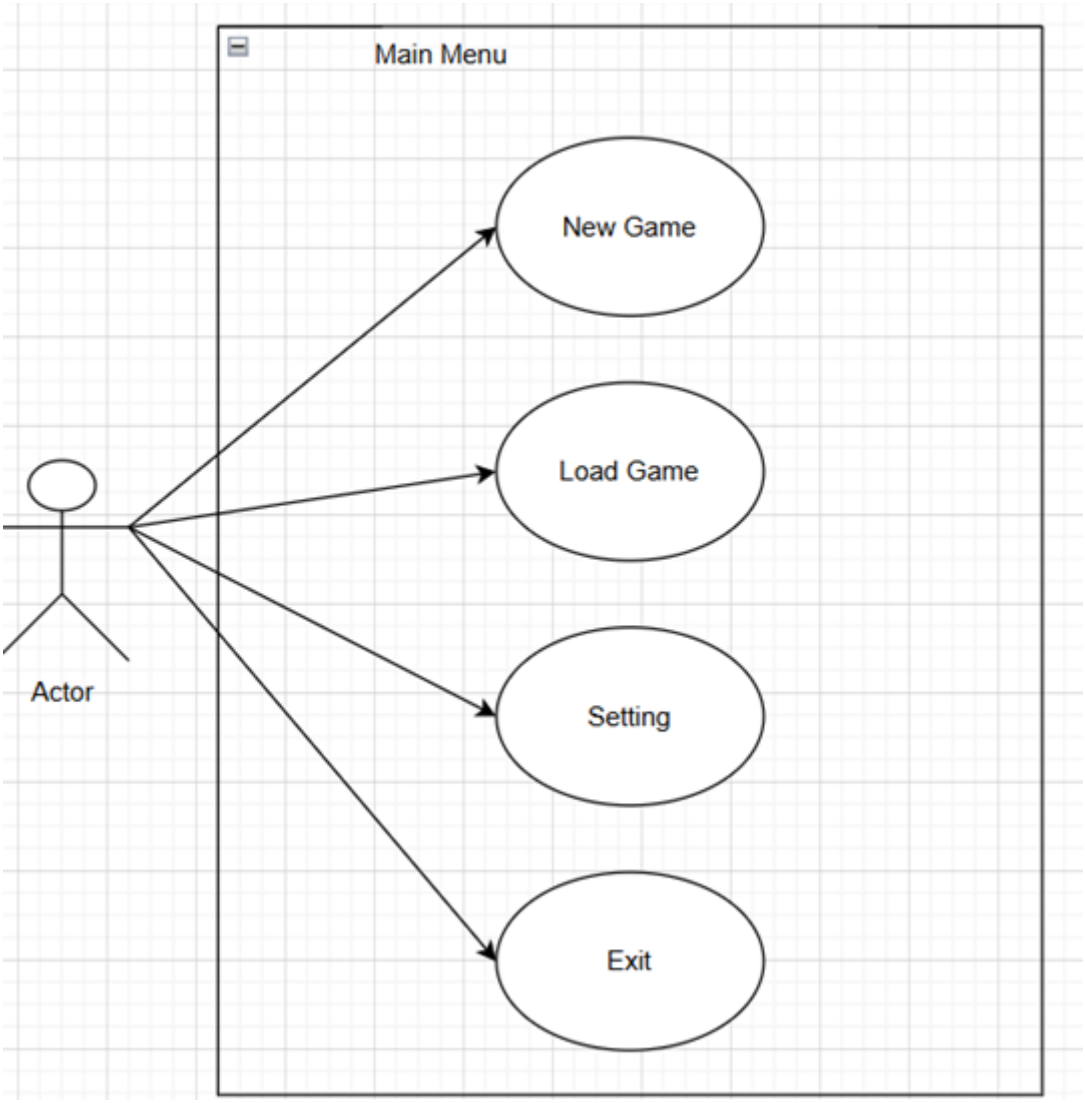


Figure 1: Main Menu Use Case [1]

3.2.5 Gameplay Use Case

3.2.5.1 Brief Description:

Players utilize Resource Data to access essential information about critical resources like oxygen, energy, and food, guiding their strategic actions. Colonist Data provides details on the health, morale, and productivity of each colonist aboard the spaceship. Specialized Systems Data offers insights into the status of vital systems like oxygen generators, power grids, and food production units. The Timeline & Minimap help players track progress, locate areas of interest, and identify threats in the environment. The Stop-Start feature enables players to control the pace of gameplay, pausing during critical decision-making or real-life interruptions. In-Game Settings offer customization options similar to the main menu, allowing players to adjust graphics, sound, and controls.

Interactive buttons such as Resource Management, Construction, and Defense provide targeted menus for managing resources, building systems, and organizing the spaceship's defense mechanisms. The Abilities Button allows players to activate special abilities, such as boosting colonist morale or temporarily increasing resource production. The Selected Button Option serves as a confirmation interface, showing the player's choices for actions or upgrades before finalizing them.

Additionally, players can engage in mini-games during specific events, such as repairing systems or countering hostile mutant attacks. Successful completion of these mini-games provides strategic advantages, such as temporarily enhanced defenses or bonus resources, helping players stay ahead of challenges.

3.2.5.2 Initial Step-by-Step Description:

Resource Data: Players can review Resource Data to monitor critical resources like oxygen, food, and energy. This allows them to allocate resources efficiently and address shortages.

Colonist Data: Players navigate to the Colonist Data section to access specific information about each colonist's health, morale, and productivity. This helps prioritize medical care or allocate work tasks.

Specialized Systems Data: Users examine Specialized Systems Data to assess the operational status of key spaceship systems, such as oxygen generators, energy grids, and food production units. Malfunctions or inefficiencies are highlighted for repair or upgrades.

Timeline & Minimap: The Timeline & Minimap feature enables players to track in-game progress, identify threats, and explore new areas of the spaceship or nearby objects in space. It ensures players remain aware of the overall game context.

Stop-Start: The Stop-Start functionality allows players to pause or resume gameplay, providing control over the pacing during critical moments, such as system malfunctions or combat scenarios.

In-Game Settings: Accessing the In-Game Settings menu lets players adjust parameters like graphics, sound, and resolution. Save, load, and exit options are also available for flexibility during gameplay.

Resource Management Button: Clicking on the Resource Management Button opens a menu displaying available resources and their usage. Player can reallocate resources to prioritize critical systems during shortages.

Construction Button: Selecting the Construction Button shows a list of structures and systems available for building or upgrading. Information about resource costs and benefits is provided to guide strategic decisions.

Defense Button: The Defense Button allows players to deploy or upgrade defensive systems, such as turrets, traps, or shields, to counter mutant threats. Strategies for covering vulnerable areas are emphasized.

Button Option: The Selected Button Option interface previews all chosen actions, such as resource reallocation, construction plans, or defensive measures. Players can confirm or modify their decisions before finalizing.

Mini-Games: During key events, such as system repairs or mutant attacks, players can engage in mini-games to gain advantages. For example:

• Successfully completing a repair mini-game restores a critical system faster.

• Winning a combat mini-game temporarily boosts defenses or damages attacking mutants.

• Completing an exploration mini-game unlocks bonus resources or rare upgrades.

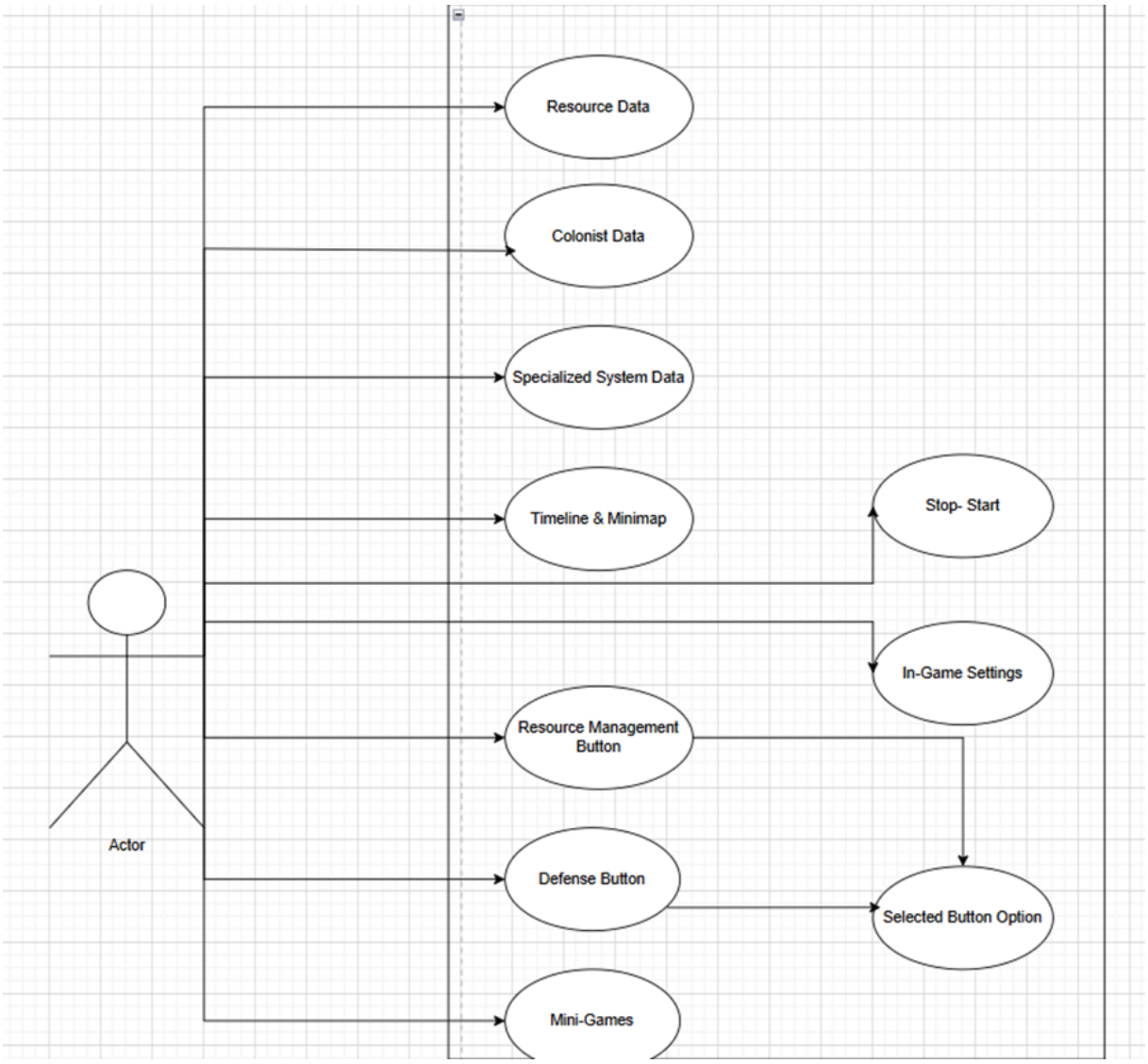


Figure 2: Gameplay Use Case [1]

3.2.6 In-Game Settings Menu Use Case

3.2.6.1 Brief Description

The use case diagram describes a settings menu where player can adjust Music Volume, Resolution, and Display Mode. They also have options to Cancel changes, Save modifications, or Quit the Game, providing a straightforward and user-friendly interface for customizing their preferences within the system.

3.2.6.2 Initial Step-by-Step Description

Music Volume: Player can adjust the background music volume to their preference.

Resolution Button: Player can modify the game's resolution to fit their PC's display capabilities.

Display Mode: Player can toggle between fullscreen and windowed modes for the game.

Cancel Button: This option allows players to discard any changes made during the session.

Save Button: This option saves all adjustments made to the settings.

Quit Game: Selecting this button takes the player back to the main menu, ending the current session.

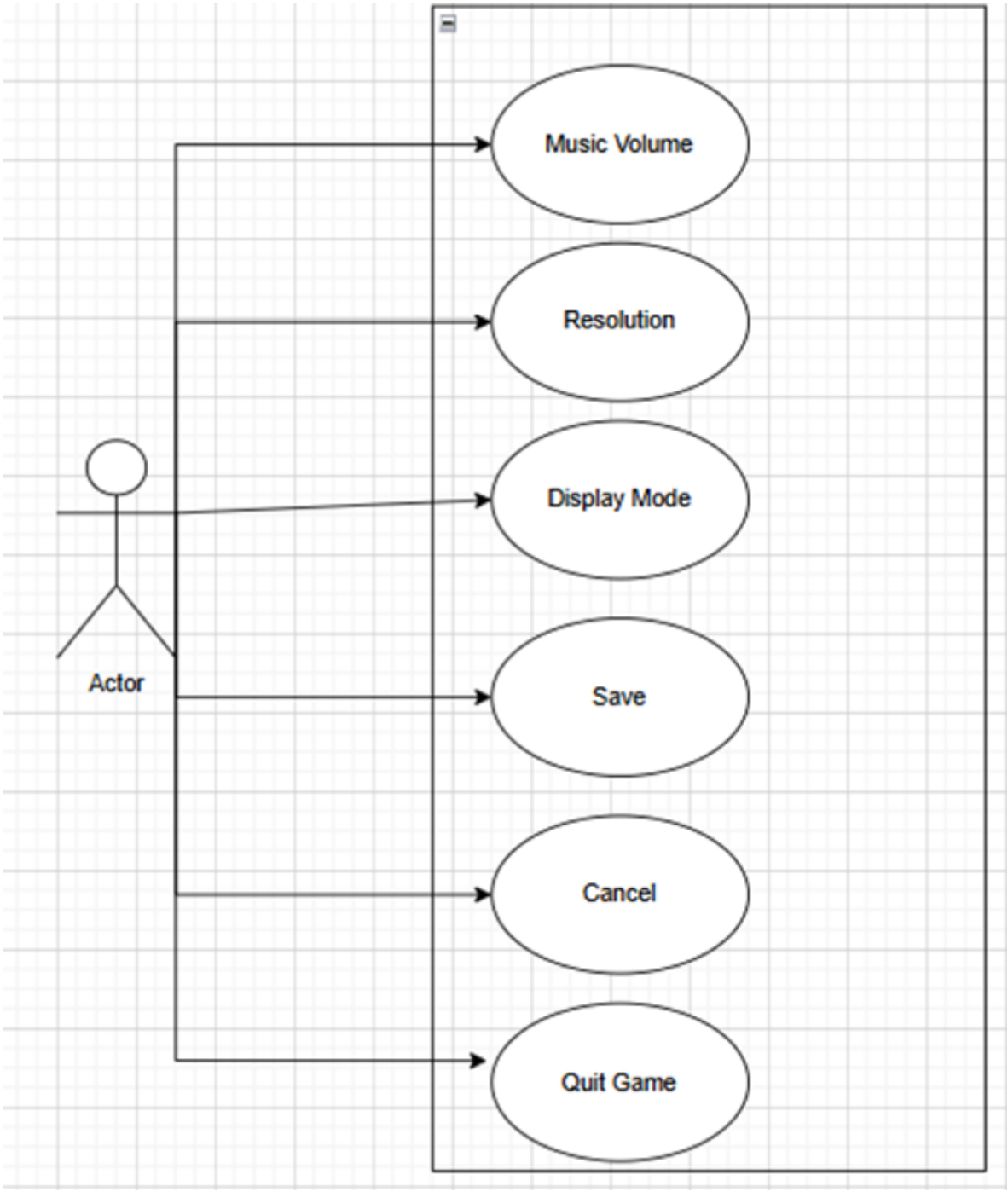


Figure 3: In-Game Settings Menu Use Case [1]

3.3 Error Handling

The system must handle errors gracefully, ensuring that any issues do not disrupt the player's experience.

• The system shall display a meaningful error message if the save file is corrupted or if there is a failure in saving or loading game data.

• The system shall provide recovery options, such as retrying or returning to the main menu in case of save/load failure.

• The system shall ensure that AI behavior remains consistent and error-free even when unexpected player actions occur.

3.4 Performance Requirements

The Spaceborn game should meet the following performance criteria to ensure a smooth gameplay experience:

The system shall load saved game data within 5 seconds.

The system shall update AI behavior for all mutants/enemies in real-time with no noticeable lag or delay.

The system shall handle up to 5 simultaneous AI agents (mutants) without affecting frame rates or gameplay performance.

Smooth and latency-free simulation visuals are crucial for maintaining a high level of immersion, and achieving this depends on various aspects of the user's PC. The minimum requirements include:

OS: Windows 7 or better

Processor: Intel i5 3.1 Ghz Quad Core

Memory: 2 GB RAM

Graphics: Nvidia GTX 660 / Radeon HD 7800 or better

Recommended Display Resolution: 1440x900

3.5 Logical Database Requirements

The Spaceborn game will include a database to store key information required for gameplay, focusing on saved game data and AI behavior. The following logical database requirements apply:

Types of Information Used by Various Functions:

• Save Game Data: Player progress, inventory, completed mini-games, and unlocked resources.

• AI Behavior Data: Information on mutant/alien types, their locations, and behavior patterns.

Frequency of Use:

• Save Game Data: Accessed during save/load operations to store and retrieve player progress.

• AI Behavior Data: Accessed during gameplay to update and retrieve enemy locations and behavior decisions.

Accessing Capabilities:

• Read: Used to load saved game data and AI behavior states.

• Write: Used to update player progress,and modify AI behavior patterns as the game progresses.

Data Entities:

• Player (ID, Name, Progress)

• Mutants/Enemies (ID, Type, Location, Behavior)

Relationships:

• Player ↔ Save Game: A player's progress is saved and loaded between sessions.

• Mutants ↔ AI Behavior: Mutants have specific behaviors stored in the database based on game progress.

Integrity Constraints:

• Data Integrity: Save game data must be accurate and consistent between sessions.

• AI Behavior Integrity: Mutant/alien behavior data must align with game progression and previous interactions.

Data Retention Requirements:

• Saved Games: Retained between sessions, with each saved game lasting at least one year from the last gameplay session.

• AI Behavior Data: Retained throughout the session and adjusted based on player actions.

3.6 Design constraints

The following design constraints will be imposed on the Spaceborn project based on external standards, hardware limitations, and gameplay requirements:

• Game Engine and Machine Learning Constraints: The choice of game engine (Unreal Engine or Unity) will dictate certain design decisions, including rendering techniques, AI behavior implementation, and memory management. The machine learning models used for AI (such as for mutant behavior) must be compatible with the selected game engine, which could restrict certain algorithms or performance features.

• Hardware Limitations: The game must be optimized to run efficiently on mid-range gaming PCs. This includes managing memory usage, optimizing frame rates, and ensuring smooth gameplay on typical consumer-grade hardware. The game must also be scalable for possible future console ports, which could introduce additional hardware constraints, such as optimizing for lower RAM and CPU capacities.

3.6.1 Standards compliance

While Spaceborn will not be subject to specific industry regulatory standards, the following general compliance and best practices will be followed:

• Data Integrity Standards: The game must follow basic data integrity standards to ensure that saved games and player progress are preserved between sessions. Backup mechanisms and error handling for corrupted save files will be implemented.

• Privacy Standards for Player Data: Any player data (e.g., saved games) will be stored and handled securely, following industry standards for privacy and data protection.

• Accessibility Standards: The game will incorporate accessibility features such as customizable key bindings, colorblind modes, and subtitle options to ensure it is accessible to a wide audience.

• Logging and Debugging Standards: During development, logging standards will be followed to track errors and gameplay issues, ensuring that bugs are

logged and traced back for resolution. These logs will be stored securely and not include any personally identifiable information.

3.7 Software System Attributes

3.7.1 Reliability

The reliability of the "Spaceborn" will be established through:

• Rigorous testing of mini-games (e.g., puzzles for oxygen generation, energy restoration) to ensure they function correctly under all scenarios.

• Stress testing the procedural map generation and alien behavior to guarantee stable performance across varying gameplay conditions.

• Fail-safe mechanisms for critical components, such as preserving embryonic status even during unexpected disruptions.

• Backup systems to restore the latest gameplay state in case of crashes.

3.7.2 Availability

This To ensure high availability for "Spaceborn":

• Checkpoint System: Autosave functionality will activate at key milestones, such as after solving a mini-game or escaping alien encounters.

• Recovery: If the game unexpectedly shuts down, the recovery system will reload the most recent checkpoint seamlessly.

• Restart Capability: Allow individual systems (e.g., machine learning-driven AI modules) to restart independently in case of failure, preventing complete game downtime.

3.7.3 Security

To ensure high availability for "Spaceborn":

• Checkpoint System: Autosave functionality will activate at key milestones, such as after solving a mini-game or escaping alien encounters.

• Recovery: If the game unexpectedly shuts down, the recovery system will reload the most recent checkpoint seamlessly.

• Restart Capability: Allow individual systems (e.g., machine learning-driven AI modules) to restart independently in case of failure, preventing complete game downtime.

3.7.4 Maintainability

Spaceborn will be designed for ease of maintenance through:

• Modular Architecture: Each system(mini games, procedural map generation, and AI behavior) will function as independent modules to simplify debugging and updates.

• Documentation: Comprehensive documentation for the C++, Python, and Unreal Engine components will ensure maintainability.

• Standardized Coding Practices: Adhere to best practices in both C++ and Python, utilizing design patterns and consistent naming conventions.

• Version Control: Use GitHub for source code versioning, enabling efficient rollback and collaborative development.

3.7.5 Portability

The portability of "Spaceborn" will ensure performance on a range of devices: • Proven Tools: Utilize Unreal Engine and its cross-platform capabilities for seamless deployment on PC and future console support.

• Language Choices: Use C++ and Python for platform-independent development.

• Optimization: Prioritize optimization techniques, such as efficient asset loading and lightweight AI computations, to ensure smooth performance even on mid-range PCs.

Software Design Description (SDD)

1.Introduction

1.1 Purpose:

This document aims to provide a comprehensive overview of the "Spaceborn" project’s software design, ensuring clarity on the system's architecture, technical components, and data flow. It serves as a technical guide for developers, testers, and stakeholders to understand the design choices, implementation details, and integration points across the system.

1.2 Definitions:

The Spaceborn project is a space-themed survival and colony management game where players manage resources, build systems, and protect their colony

from external threats. The game incorporates procedural map generation, AI-driven characters, and dynamic events. The scope includes:

- Procedural Map and Event Generation
- AI for NPC Behavior and Combat Systems
- Resource and Colony Management
- Immersive Survival Gameplay with Strategic Elements.

1.3 Overview

This document covers the following areas:

- System Architecture: Describes the software layers and component interactions.
- Data Management Strategies: Defines database design, data security, and backup policies.
- Hardware and Software Requirements: Details system prerequisites for optimal performance.
- System Interfaces: Describes how users and other systems interact with Spaceborn.
- Scalability and Performance Requirements: Ensures the system can grow and meet performance benchmarks.
- Security Protocols: Protects user data and maintains secure communication.

1.4 Glossary

Term	Definition
UI:	User Interface
AI:	Artificial Intelligence
FPS:	Frames Per Second
HUD	Heads-Up Display

2. System Overview:

2.1 System Objectives

The primary objectives of the Spaceborn system include:

- Creating an immersive colony-building and survival experience.
- Utilizing procedural generation to ensure replayability.
- Integrating AI-driven NPC behavior and events.
- Ensuring high performance across multiple platforms (PC, VR, consoles).

2.2 System Architecture

Spaceborn follows a layered architecture, dividing functionality into the following layers:

1. Presentation Layer: User interface elements, HUD, and menu systems.
2. Application Layer: Game logic, resource management, combat systems, AI behavior.
3. Data Layer: Database interactions, game state storage, and inventory management.
4. Networking Layer (Future Scope): Multiplayer synchronization and communication.

3. Data Design

3.1 Database Design

The game uses a relational database system to manage persistent and transient data.

Key Tables:

- Users: Stores player profiles, credentials, achievements.
- Inventory: Tracks collected resources and crafted items.

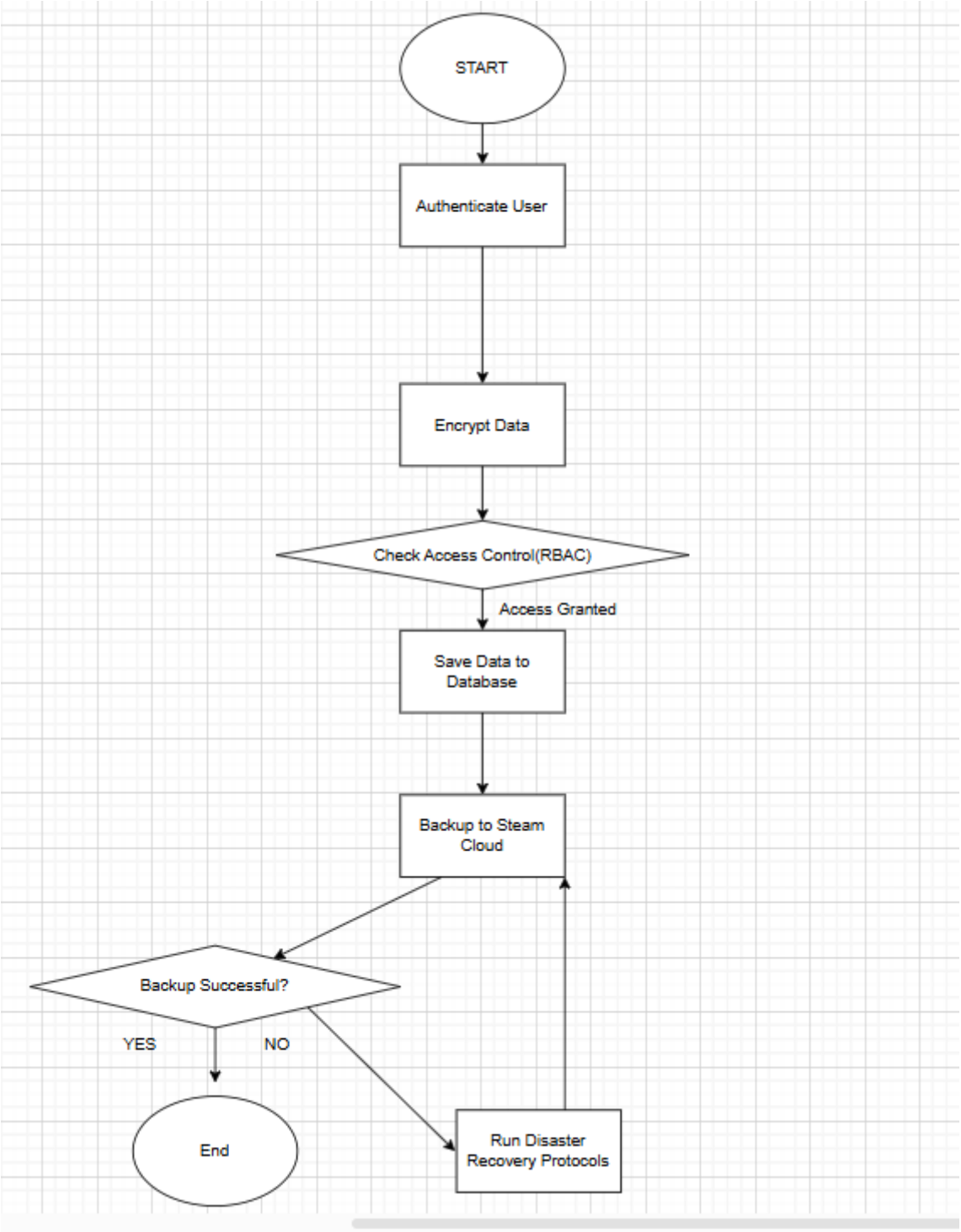
- GameState: Records checkpoints, mission progress, and environmental states.
- NPCState: Manages NPC health, status, and behavior triggers.

3.2 Data Security

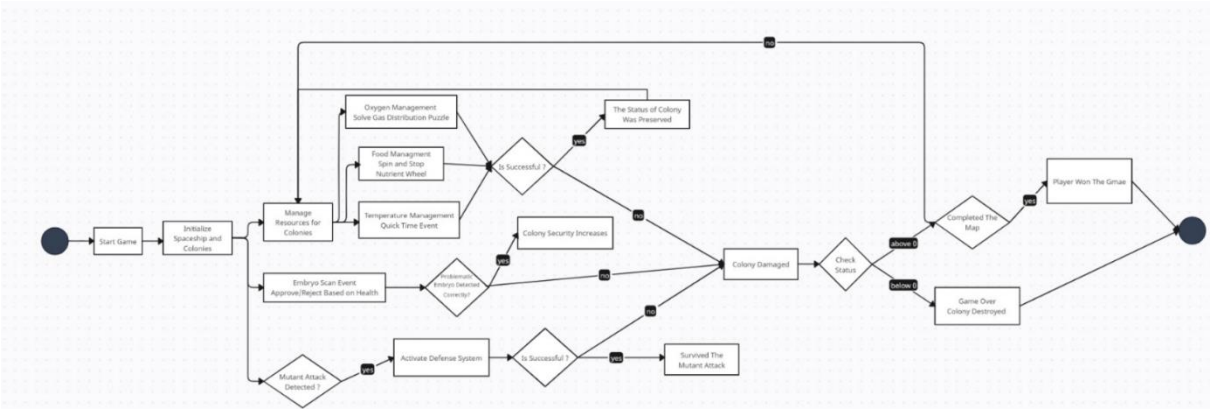
- Authorization and Authentication: Only authorized persons can access player data.
- Backup: Automated cloud backups using Steam Cloud API.
- Access Control: RBAC (Role-Based Access Control) to limit unauthorized access.

3.3 Data Backup and Recovery

- Incremental data backups reduce resource load.
- Disaster recovery protocols ensure minimal data loss in failures.

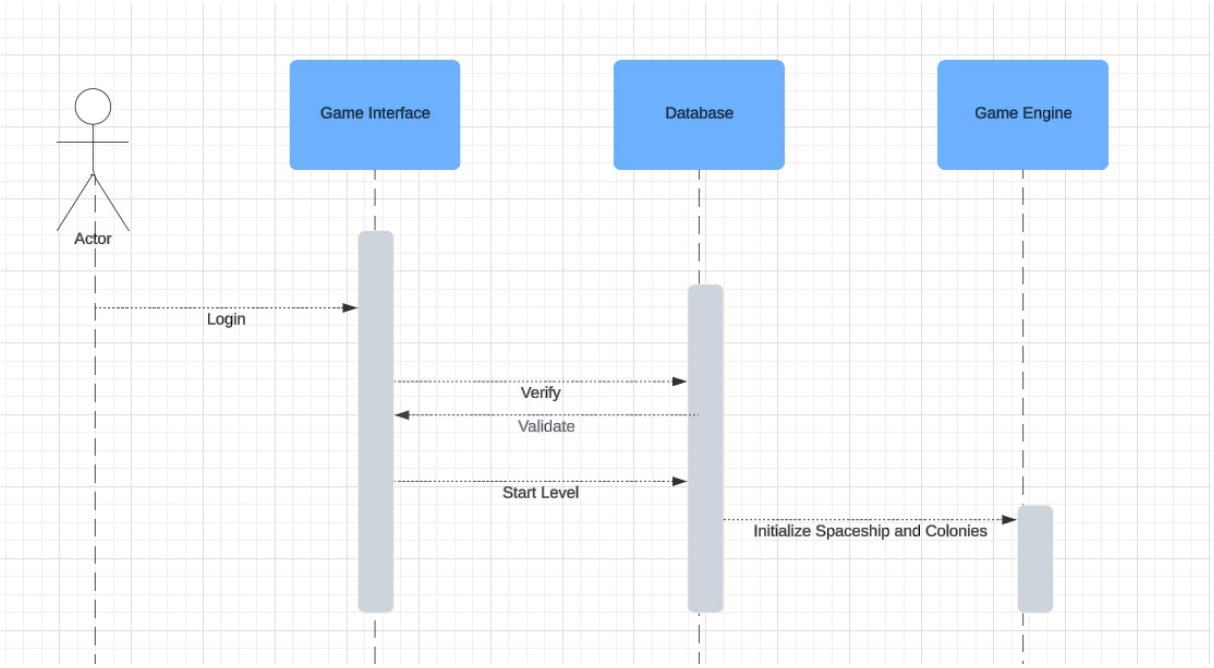

START
Authenticate User
Encrypt Data
Check Access Control(RBAC)
Access Granted
Save Data to Database
Backup to Steam Cloud
Backup Successful?
YES
NO
End
Run Disaster Recovery Protocols

3.4 Activity Diagram

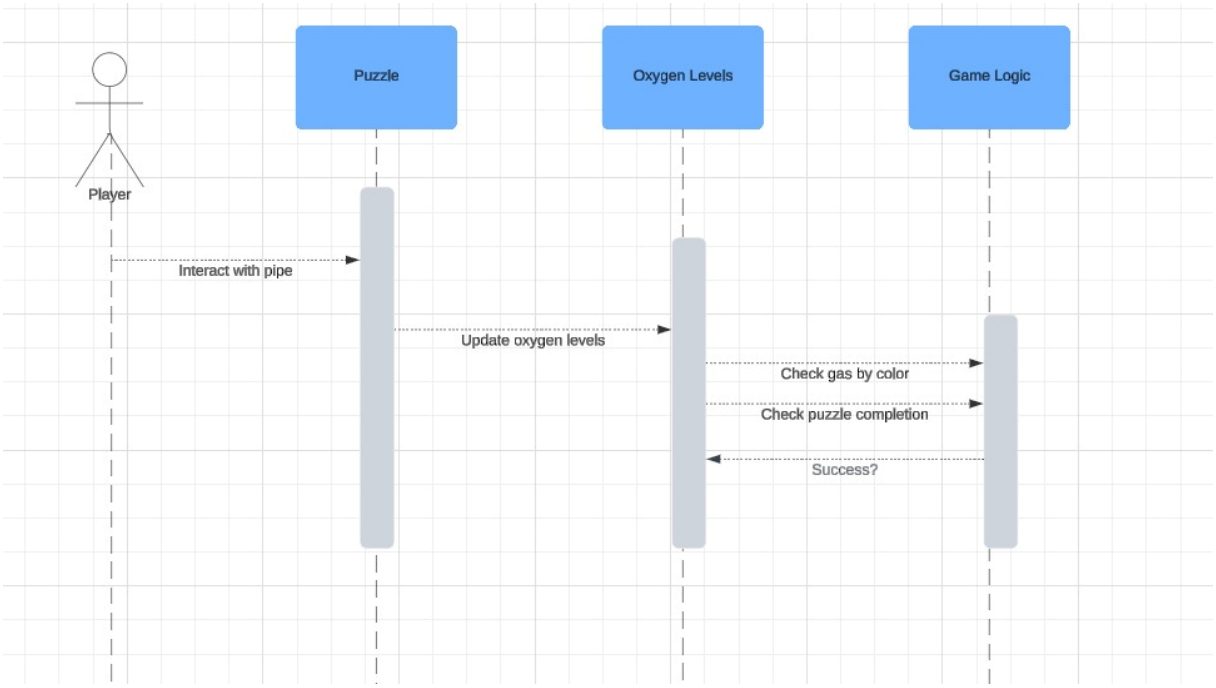


3.5 Sequence Diagram

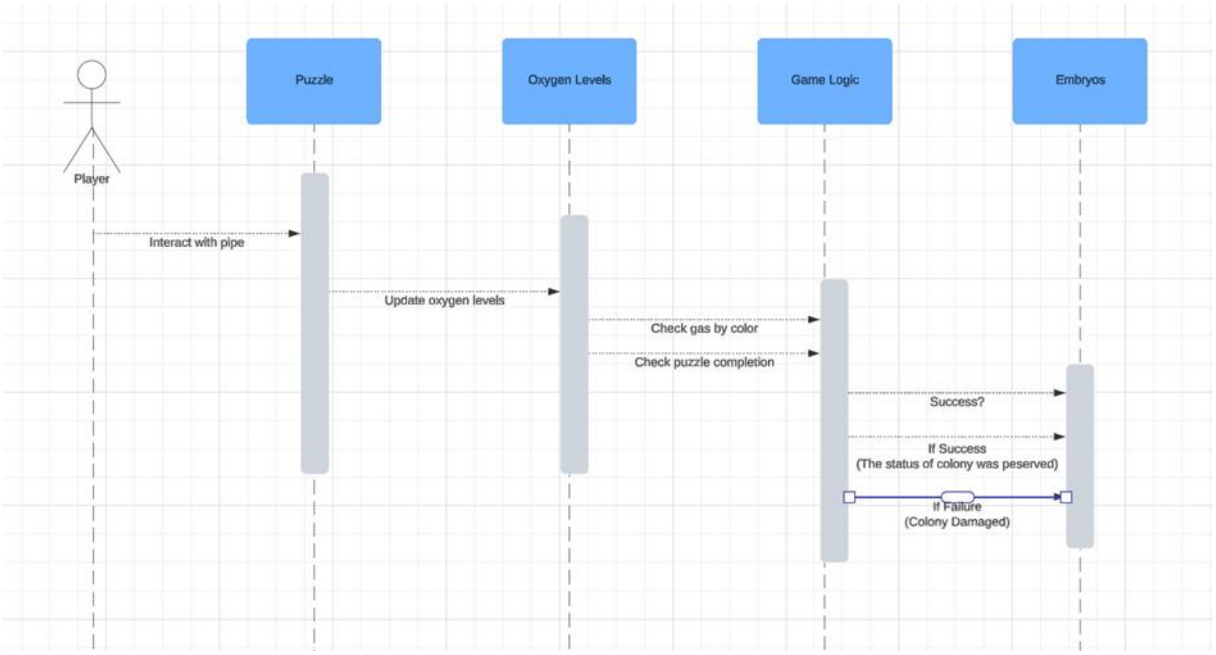
3.5.1 Login and Initialize Game



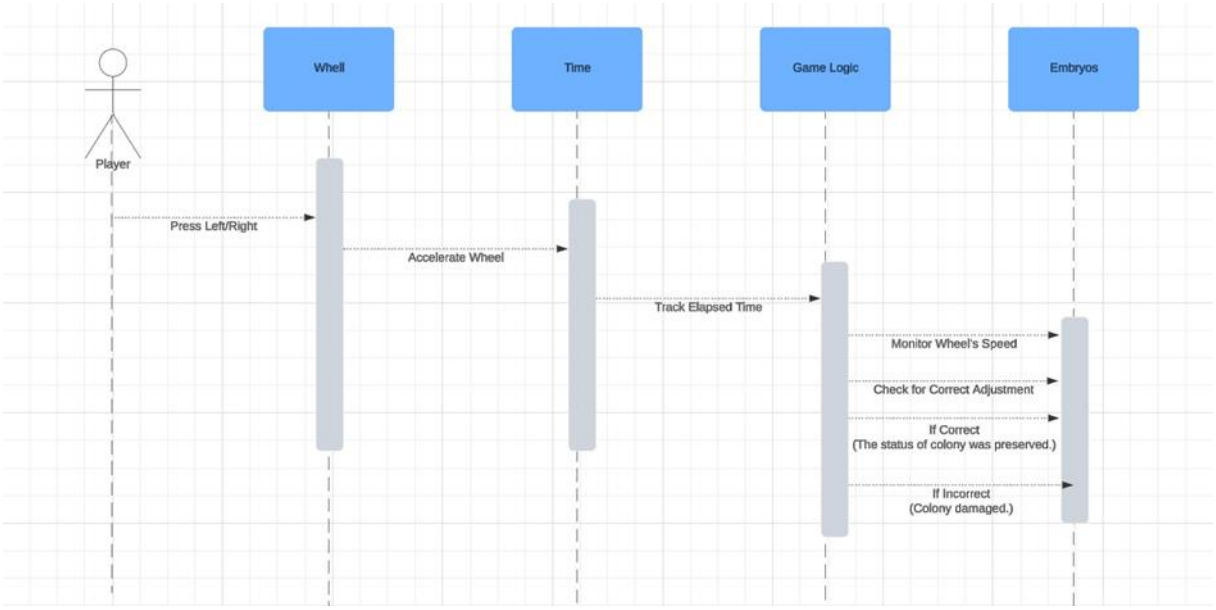
3.5.2 Oxygen Management Puzzle



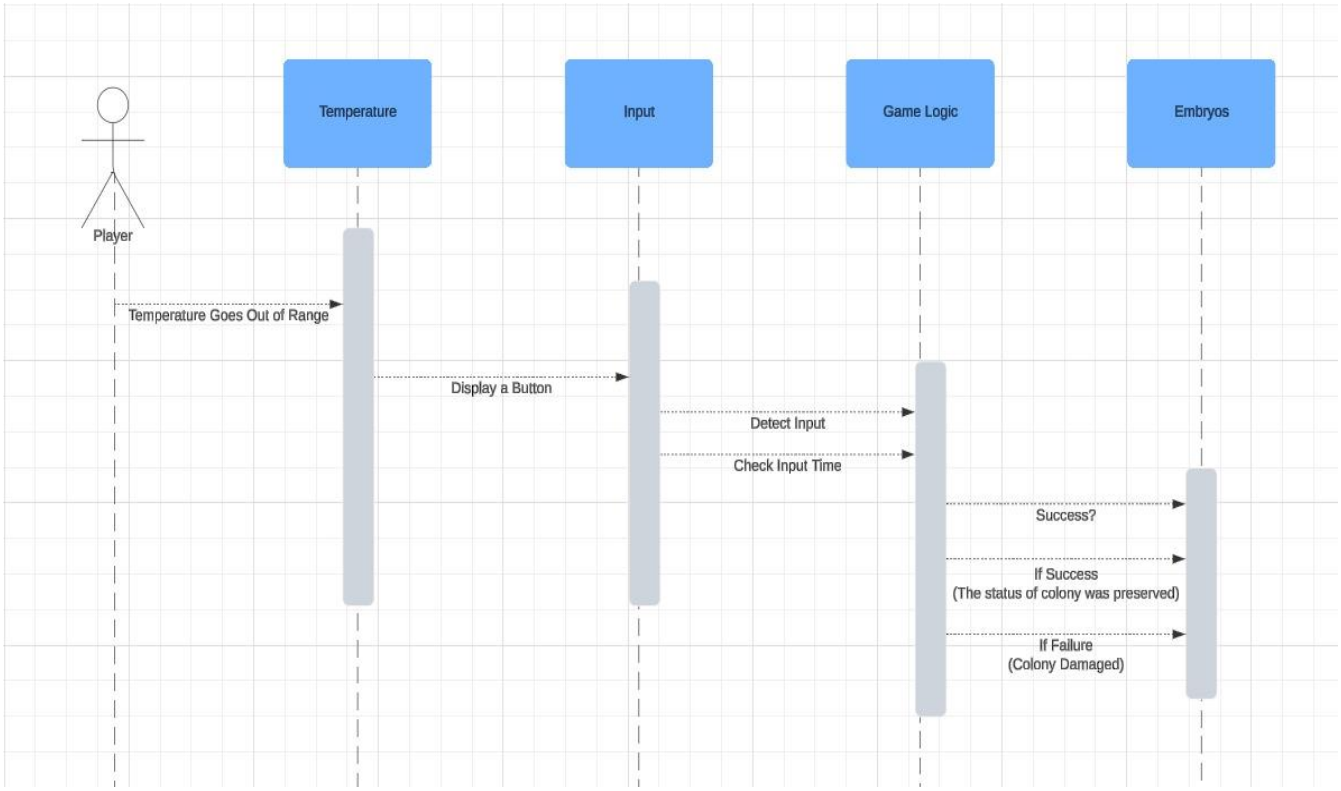
3.5.3 Oxygen Management and Colony Status



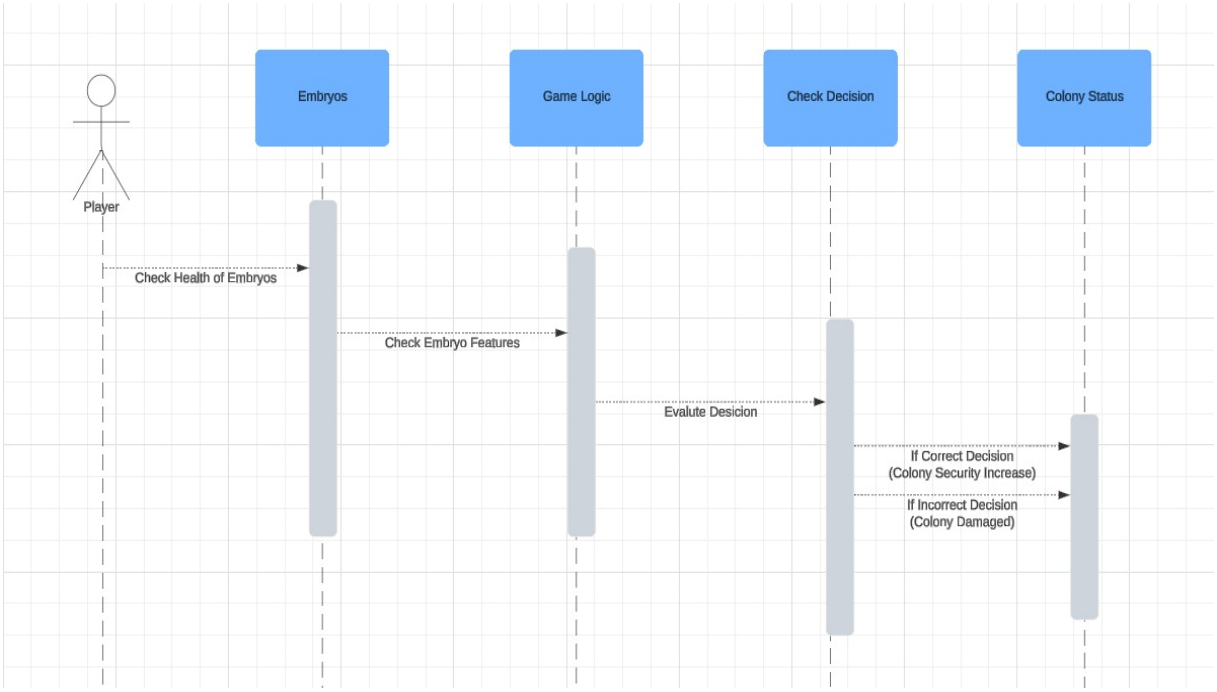
3.5.4 Food Management Wheel



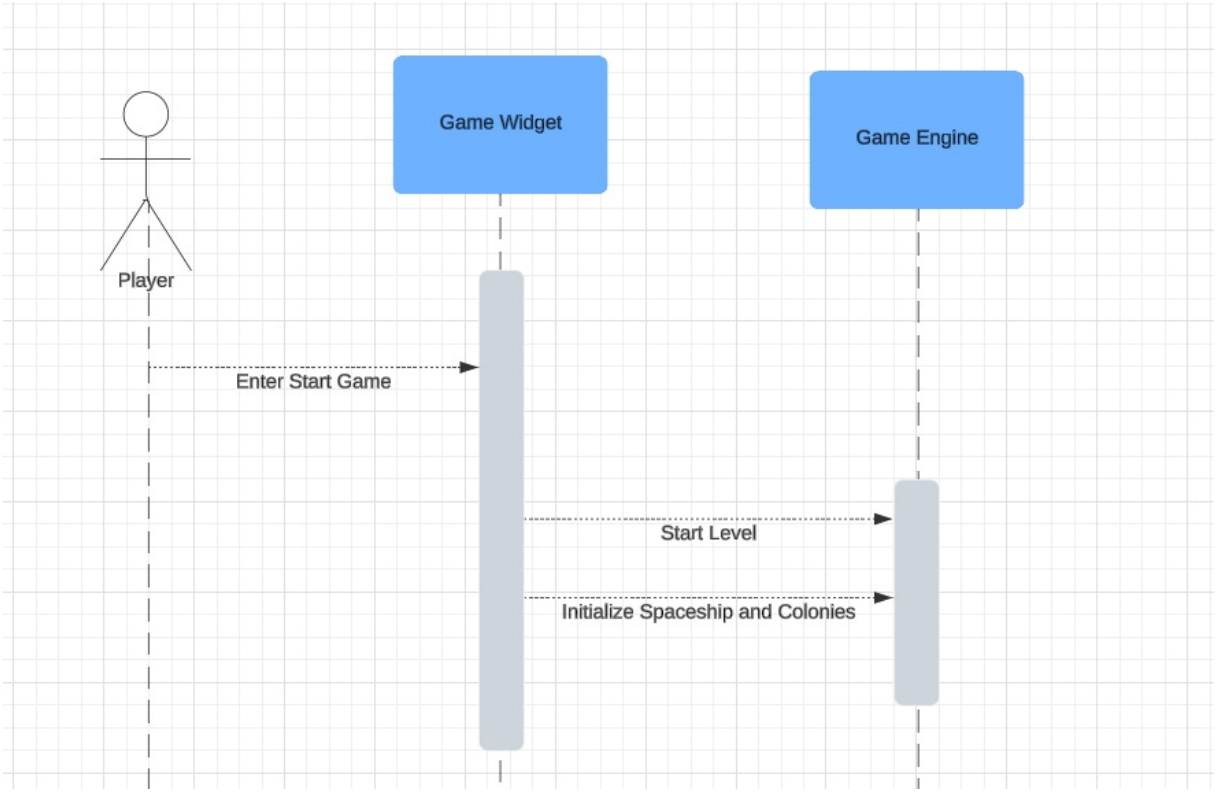
3.5.5 Temperature Management



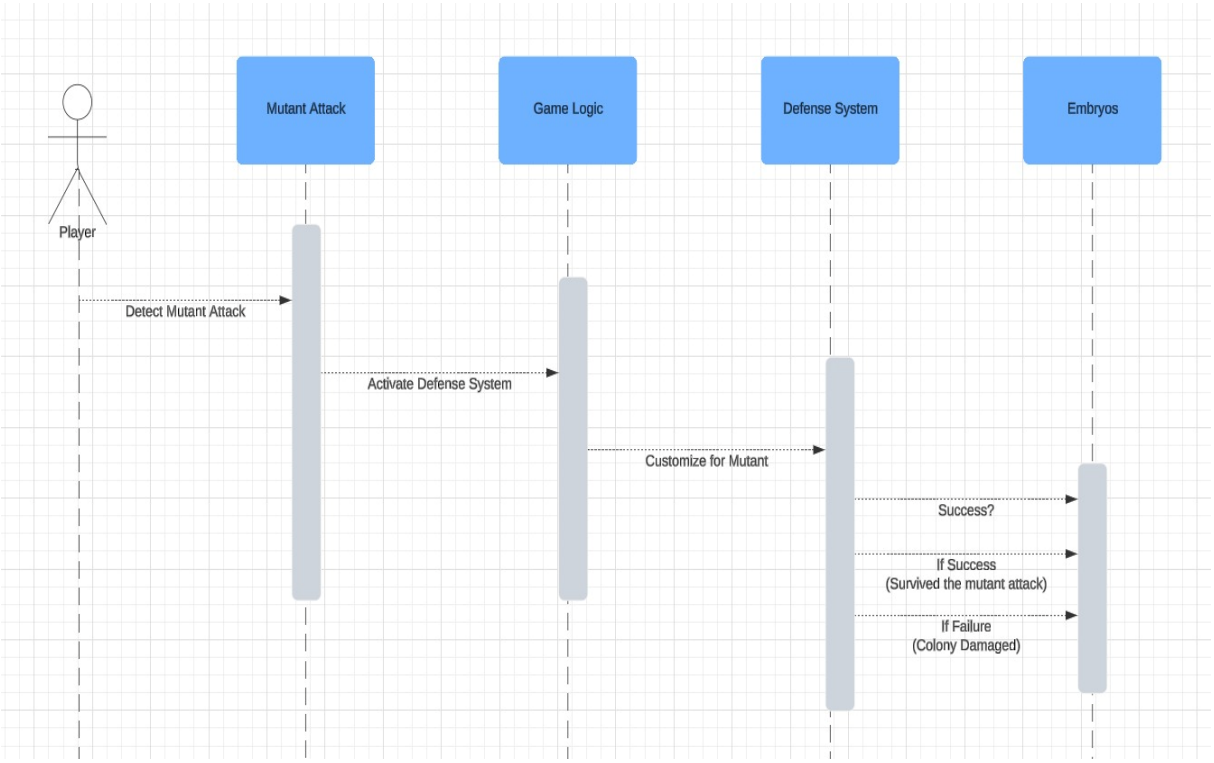
3.5.6 Embryo Health Check



3.5.7 Start Game Sequence



3.5.8 Mutant Attack Scenario



4. Interface Design

This section describes the basic interfaces of Spaceborn and the purpose of each. The interfaces are designed to allow players to log in to the game, manage their colonies, and view important information.

Login Page

Description:

This is the page where players perform their identity verification.

Username and password are entered.

Player progress, achievements and game preferences are loaded from the database.

Key Features:

Username and password input boxes.

"Log In" and "Forgot My Password" buttons.

Game version and update notes information.

Purpose: To provide players with safe and fast access to the game.

Simulation Start Page:

Description:

This is the page where the player makes the necessary settings to start the spaceship and colonies.

Players can choose the game difficulty level and starting resources.

Colony and spaceship systems are started here.

Basic Features:

Difficulty level selection (Easy, Medium, Hard).

Editing starting resources (Oxygen, Food, Energy).

"Start" button.

Purpose: To prepare the player for the game by making strategic starting settings.

Simulation View Page

Description:

This is the page where the player makes the necessary settings to start the spaceship and colonies.

Players can choose the game difficulty level and starting resources.

Colony and spaceship systems are started here.

Basic Features:

Editing starting resources (Oxygen, Food, Energy).

"Start" button.

Purpose: To prepare the player for the game by making strategic starting settings.

Point Cloud View Page

Description:

This is the page where the environmental data scanned by the spaceship's sensors is visualized for the players.

The data obtained during exploration is shown here in the form of point clouds.

The player can analyze threats or undiscovered resources on this page.

Key Features:

3D Point Cloud Map.

Zoom and Pan options.

Threat and resource markers.

Purpose: To enable the player to make strategic decisions by analyzing exploration data.

References

The resources used to produce this document should be listed here and referenced in the relevant text of this document

[1] Software Engineeering Department, "Graduation Projects," SENG, [Online]. Available: https://seng.cankaya.edu.tr/graduation-projects/. [Accessed 28 June 2024].

https://bilgipaketi.cankaya.edu.tr/CourseInfo?Id=105857&BolumKodu=701&MNo=387

https://catnessgames.com/blog/unreal-engine-5-requirements/